Configuring TCP

TCP is a protocol that specifies the format of data and acknowledgments used in data transfer. TCP is a connection-oriented protocol because participants must establish a connection before data can be transferred. By performing flow control and error correction, TCP guarantees reliable, in-sequence delivery of packets. TCP is considered a reliable protocol because it will continue to request an IP packet that is dropped or received out of order until it is received. This module explains concepts related to TCP and how to configure TCP in a network.

Prerequisites for TCP

TCP Time Stamp, TCP Selective Acknowledgment, and TCP Header Compression

Because TCP time stamps are always sent and echoed in both directions and the time-stamp value in the header is always changing, TCP header compression will not compress the outgoing packet. To allow TCP header compression over a serial link, the TCP time-stamp option is disabled. If you want to use TCP header compression over a serial line, TCP time stamp and TCP selective acknowledgment must be disabled. Both features are disabled by default. Use the **no ip tcp selective-ack** command to disable the TCP selective acknowledgment once it is enabled.

Information About TCP

TCP Services

TCP provides reliable transmission of data in an IP environment. TCP corresponds to the transport layer (Layer 4) of the Open Systems Interconnection (OSI) reference model. Among the services that TCP provides are stream data transfer, reliability, efficient flow control, full-duplex operation, and multiplexing.

With stream data transfer, TCP delivers an unstructured stream of bytes that are identified by sequence numbers. This service benefits applications because they do not have to divide data into blocks before handing it off to TCP. Instead, TCP groups bytes into segments and passes them to IP for delivery.

TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through an internetwork. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte that the source expects to receive. Bytes that are not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to handle lost, delayed, duplicate, or misread packets. A timeout mechanism allows devices to detect lost packets and request retransmission.

TCP offers efficient flow control, which means that the receiving TCP process indicates the highest sequence number that it can receive without overflowing its internal buffers when sending acknowledgments back to the source.

TCP offers full-duplex operation, and TCP processes can both send and receive data at the same time.

TCP multiplexing allows numerous simultaneous upper-layer conversations to be multiplexed over a single connection.

TCP Connection Establishment

To use reliable transport services, TCP hosts must establish a connection-oriented session with one another. Connection establishment is performed by using a “three-way handshake” mechanism.

A three-way handshake synchronizes both ends of a connection by allowing both sides to agree upon the initial sequence numbers. This mechanism guarantees that both sides are ready to transmit data. The three-way handshake is necessary so that packets are not transmitted or retransmitted during session establishment or after session termination.

Each host randomly chooses a sequence number, which is used to track bytes within the stream that the host is sending. The three-way handshake proceeds in the following manner:

- The first host (Host A) initiates a connection by sending a packet with the initial sequence number (X) and the synchronize/start (SYN) bit set to indicate a connection request.
- The second host (Host B) receives the SYN, records the sequence number X, and replies by acknowledging (ACK) the SYN (with an ACK = X + 1). Host B includes its own initial sequence number (SEQ = Y). An ACK = 20 means that the host has received bytes 0 through 19 and expects byte 20 next. This technique is called forward acknowledgment.
- Host A acknowledges all bytes that Host B has sent with a forward acknowledgment indicating the next byte Host A expects to receive (ACK = Y + 1). Data transfer can then begin.

TCP Connection Attempt Time

You can set the amount of time the software will wait before attempting to establish a TCP connection. The connection attempt time is a host parameter and pertains to traffic that originated at the device and not to traffic going through the device. To set the TCP connection attempt time, use the **ip tcp synwait-time** command in global configuration mode. The default is 30 seconds.

TCP Selective Acknowledgment

The TCP Selective Acknowledgment feature improves performance if multiple packets are lost from one TCP window of data.

Prior to this feature, because of limited information available from cumulative acknowledgments, a TCP sender could learn about only one lost packet per-round-trip time. An aggressive sender could choose to resend packets early, but such re-sent segments might have already been successfully received.

The TCP selective acknowledgment mechanism helps improve performance. The receiving TCP host returns selective acknowledgment packets to the sender, informing the sender of data that has been received. In other words, the receiver can acknowledge packets received out of order. The sender can then resend only missing data segments (instead of everything since the first missing packet).

Prior to selective acknowledgment, if TCP lost packets 4 and 7 out of an 8-packet window, TCP would receive acknowledgment of only packets 1, 2, and 3. Packets 4 through 8 would need to be re-sent. With selective acknowledgment, TCP receives acknowledgment of packets 1, 2, 3, 5, 6, and 8. Only packets 4 and 7 must be re-sent.

TCP selective acknowledgment is used only when multiple packets are dropped within one TCP window. There is no performance impact when the feature is enabled but not used. Use the **ip tcp selective-ack** command in global configuration mode to enable TCP selective acknowledgment.

Refer to RFC 2018 for more details about TCP selective acknowledgment.

TCP Time Stamp

The TCP time-stamp option provides improved TCP round-trip time measurements. Because the time stamps are always sent and echoed in both directions and the time-stamp value in the header is always changing, TCP header compression will not compress the outgoing packet. To allow TCP header compression over a serial link, the TCP time-stamp option is disabled. Use the **ip tcp timestamp** command to enable the TCP time-stamp option.

Refer to RFC 1323 for more details on TCP time stamps.

TCP Maximum Read Size

The maximum number of characters that TCP reads from the input queue for Telnet and relogin at one time is very large (the largest possible 32-bit positive number) by default. To change the TCP maximum read size value, use the **ip tcp chunk-size** command in global configuration mode.

Note We do not recommend that you change this value.

TCP Path MTU Discovery

Path MTU Discovery is a method for maximizing the use of the available bandwidth in the network between endpoints of a TCP connection, which is described in RFC 1191. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable maximum transmission unit (MTU) size of the various links along the path. Sometimes a device is unable to forward a datagram because it requires fragmentation (the packet is larger than the MTU that you set for the interface with the **interface** configuration

command), but the “do not fragment” (DF) bit is set. The intermediate gateway sends a “Fragmentation needed and DF bit set” Internet Control Message Protocol (ICMP) message to the sending host, alerting the host to the problem. On receiving this message, the host reduces its assumed path MTU and consequently sends a smaller packet that will fit the smallest packet size of all links along the path.

By default, TCP Path MTU Discovery is disabled. Existing connections are not affected irrespective of whether this feature is enabled or disabled.

Customers using TCP connections to move bulk data between systems on distinct subnets would benefit most by enabling this feature. Customers using remote source-route bridging (RSRB) with TCP encapsulation, serial tunnel (STUN), X.25 Remote Switching (also known as XOT or X.25 over TCP), and some protocol translation configurations might also benefit from enabling this feature.

Use the **ip tcp path-mtu-discovery** global configuration command to enable Path MTU Discovery for connections initiated by the device when the device is acting as a host.

For more information about Path MTU Discovery, refer to the “Configuring IP Services” module of the *IP Application Services Configuration Guide*.

TCP Window Scaling

The TCP Window Scaling feature adds support for the Window Scaling option in RFC 1323,*TCP Extensions for High Performance*. A larger window size is recommended to improve TCP performance in network paths with large bandwidth-delay product characteristics that are called Long Fat Networks (LFNs). The TCP Window Scaling enhancement provides LFN support.

The window scaling extension expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32-bit value in the 16-bit window field of the TCP header. The window size can increase to a scale factor of 14. Typical applications use a scale factor of 3 when deployed in LFNs.

The TCP Window Scaling feature complies with RFC 1323. The maximum window size was increased to 1,073,741,823 bytes. The larger scalable window size will allow TCP to perform better over LFNs. Use the **ip tcp window-size** command in global configuration mode to configure the TCP window size.

TCP Sliding Window

A TCP sliding window provides an efficient use of network bandwidth because it enables hosts to send multiple bytes or packets before waiting for an acknowledgment.

In TCP, the receiver specifies the current window size in every packet. Because TCP provides a byte-stream connection, window sizes are expressed in bytes. A window is the number of data bytes that the sender is allowed to send before waiting for an acknowledgment. Initial window sizes are indicated at connection setup, but might vary throughout the data transfer to provide flow control. A window size of zero means “Send no data.” The default TCP window size is 4128 bytes. From Cisco IOS XE 17.14.1a, the default TCP window size is 131072 bytes. Use the ip tcp window-size command to change the default window size.

Note In the presence of optimizations (**ip tcp ack-tuning** and **ip tcp winupdate-opt**) which are enabled by default from 17.10.1, the value 131072 will be applied unless it is configured otherwise. Although, when default keyword is used for the **ip tcp window-size** command prior to version 17.14.1a, the value 4128 will take effect.

In a TCP sliding-window operation, for example, the sender might have a sequence of bytes to send (numbered 1 to 10) to a receiver who has a window size of five. The sender then places a window around the first five bytes and transmits them together. The sender then waits for an acknowledgment.

The receiver responds with an ACK = 6, indicating that it has received bytes 1 to 5 and is expecting byte 6 next. In the same packet, the receiver indicates that its window size is 5. The sender then moves the sliding window five bytes to the right and transmits bytes 6 to 10. The receiver responds with an ACK = 11, indicating that it is expecting sequenced byte 11 next. In this packet, if the receiver indicates that its window size is 0, the sender cannot send any more bytes until the receiver sends another packet with a window size greater than 0.

TCP Outgoing Queue Size

The default TCP outgoing queue size per connection is five segments if the connection has a TTY associated with it (such as a Telnet connection). If no TTY connection is associated with a connection, the default queue size is 20 segments. Use the **ip tcp queuemax** command to change the five-segment default value.

TCP MSS Adjustment

The TCP MSS Adjustment feature enables the configuration of the maximum segment size (MSS) for transient packets that traverse a device, specifically TCP segments with the SYN bit set. Use the **ip tcp adjust-mss** command in interface configuration mode to specify the MSS value on the intermediate device of the SYN packets to avoid truncation.

When a host (usually a PC) initiates a TCP session with a server, the host negotiates the IP segment size by using the MSS option field in the TCP SYN packet. The value of the MSS field is determined by the MTU configuration on the host. The default MSS value for a PC is 1500 bytes.

The PPP over Ethernet (PPPoE) standard supports a Maximum Transmission Unit (MTU) of only 1492 bytes. The disparity between the host and PPPoE MTU size can cause the device in between the host and the server to drop 1500-byte packets and terminate TCP sessions over the PPPoE network. Even if the path MTU (which detects the correct MTU across the path) is enabled on the host, sessions may be dropped because system administrators sometimes disable ICMP error messages that must be relayed from the host for path MTU to work.

The **ip tcp adjust-mss** command helps prevent TCP sessions from being dropped by adjusting the MSS value of the TCP SYN packets.

The **ip tcp adjust-mss** command is effective only for TCP connections passing through the device.

In most cases, the optimum value for the *max-segment-size* argument of the **ip tcp adjust-mss** command is 1452 bytes. This value plus the 20-byte IP header, the 20-byte TCP header, and the 8-byte PPPoE header add up to a 1500-byte packet that matches the MTU size for the Ethernet link.

See the “Configuring the MSS Value and MTU for Transient TCP SYN Packets” section for configuration instructions.

TCP Applications Flags Enhancement

The TCP Applications Flags Enhancement feature enables the user to display additional flags with reference to TCP applications. There are two types of flags: status and option. The status flags indicate the status of TCP connections such as passive open, active open, retransmission timeout, and app closed for listening. The additional flags indicate the state of set options such as whether a VPN routing and forwarding instance (VRF)

is set, whether a user is idle, and whether a keepalive timer is running. Use the **show tcp** command to display TCP application flags.

TCP Show Extension

The TCP Show Extension feature introduces the capability to display addresses in IP format instead of the hostname format and to display the VRF table associated with the connection. To display the status for all endpoints with addresses in IP format, use the **show tcp brief numeric** command.

TCP MIB for RFC 4022 Support

The TCP MIB for RFC 4022 Support feature introduces support for RFC 4022, *Management Information Base for the Transmission Control Protocol (TCP)*. RFC 4022 is an incremental change of the TCP MIB to improve the manageability of TCP.

To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:

http://www.cisco.com/go/mibs

Zero-Field TCP Packets

Prior to Cisco IOS XE Release 2.5, when a zero-field TCP packet is received on the router, the TCP packet counter is incremented.

In Cisco IOS XE Release 2.5 and later releases, when a zero-field TCP packet is received on the router, the TCP packet counter is not incremented.

When a zero-field TCP packet is received, it is displayed as 0 under the TCP statistics field when the **show ip traffic** command is configured. When the debug **ip tcp packet** command is configured, and a zero-field TCP packet is received, a debug message similar to the following is displayed:

```
Jan 19 21:57:28.487: TCP: Alert! Received a segment with cleared flags
10.4.14.49
```

How to Configure TCP

Configuring TCP Performance Parameters

Before you begin

Both sides of the network link must be configured to support window scaling or the default of 65,535 bytes will be applied as the maximum window size. To support Explicit Congestion Notification (ECN), the remote peer must be ECN-enabled because the ECN capability is negotiated during a three-way handshake with the remote peer.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **ip tcp synwait-time** *seconds*
4. **ip tcp path-mtu-discovery** [**age-timer** {*minutes* | **infinite**}]
5. **ip tcp selective-ack**
6. **ip tcp timestamp**
7. **ip tcp chunk-size** *characters*
8. **ip tcp window-size** *bytes*
9. **ip tcp ecn**
10. **ip tcp queuemax** *packets*
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip tcp synwait-time** *seconds* **Example:** `Device(config)# ip tcp synwait-time 60`	(Optional) Sets the amount of time the Cisco software will wait before attempting to establish a TCP connection. • The default is 30 seconds.
Step 4	**ip tcp path-mtu-discovery** [**age-timer** {*minutes* | **infinite**}] **Example:** `Device(config)# ip tcp path-mtu-discovery age-timer 11`	(Optional) Enables Path MTU Discovery. • **age-timer** —Time interval, in minutes, TCP reestimates the Maximum Transmission Unit (MTU) with a larger Maximum Segment Size (MSS). The default is 10 minutes. The maximum is 30 minutes. • **infinite**—Disables the age timer.
Step 5	**ip tcp selective-ack** **Example:** `Device(config)# ip tcp selective-ack`	(Optional) Enables TCP selective acknowledgment.
Step 6	**ip tcp timestamp** **Example:** `Device(config)# ip tcp timestamp`	(Optional) Enables the TCP time stamp.
Step 7	**ip tcp chunk-size** *characters* **Example:**	(Optional) Sets the TCP maximum read size for Telnet or rlogin.

	Command or Action	Purpose
	`Device(config)# ip tcp chunk-size 64000`	**Note** We do not recommend that you change this value.
Step 8	**ip tcp window-size** *bytes* **Example:** `Device(config)# ip tcp window-size 75000`	(Optional) Sets the TCP window size. • The bytes argument can be set to an integer from 68 to 1073741823. To enable window scaling to support Long Flat Networks (LFNs), the TCP window size must be more than 65535. The default window size from Cisco IOS XE 17.14.1a is 131072 if window scaling is not configured. Although, when default keyword is used for the **ip tcp window-size** command prior to version 17.14.1a, the value 4128 will take effect if window scaling is not configured. **Note** With CSCsw45317, the *bytes* argument can be set to an integer from 68 to 1073741823.
Step 9	**ip tcp ecn** **Example:** `Device(config)# ip tcp ecn`	(Optional) Enables ECN for TCP.
Step 10	**ip tcp queuemax** *packets* **Example:** `Device(config)# ip tcp queuemax 10`	(Optional) Sets the TCP outgoing queue size.
Step 11	**end** **Example:** `Device(config)# end`	Exits to privileged EXEC mode.

Configuring the MSS Value and MTU for Transient TCP SYN Packets

Perform this task to configure the maximum size segment (MSS) for transient packets that traverse a device, specifically TCP segments with the SYN bit set, and to configure the MTU size of IP packets.

If you are configuring the **ip mtu** command on the same interface as the **ip tcp adjust-mss** command, we recommend that you use the following commands and values:

- **ip tcp adjust-mss 1452**
- **ip mtu 1492**

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip tcp adjust-mss** *max-segment-size*

5. **ip mtu** *bytes*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/0`	Configures an interface type and enters interface configuration mode.
Step 4	**ip tcp adjust-mss** *max-segment-size* **Example:** `Device(config-if)# ip tcp adjust-mss 1452`	Adjusts the MSS value of TCP SYN packets going through a device. • The *max-segment-size* argument is the maximum segment size, in bytes. The range is from 500 to 1460.
Step 5	**ip mtu** *bytes* **Example:** `Device(config-if)# ip mtu 1492`	Sets the MTU size of IP packets, in bytes, sent on an interface.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits to global configuration mode.

Configuring the MSS Value for IPv6 Traffic

Perform this task to configure the maximum size segment (MSS) for transient packets that traverse a device, specifically TCP segments with the DF bit set in IPv6 network layer (IP) header.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 tcp adjust-mss** *max-segment-size*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/0`	Configures an interface type and enters interface configuration mode.
Step 4	**ipv6 tcp adjust-mss** *max-segment-size* **Example:** `Device(config-if)# ipv6 tcp adjust-mss 1452`	Adjusts the MSS value of TCP DF packets going through a device. • The *max-segment-size* argument is the maximum segment size, in bytes. The range is from 40 to 1940.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Verifying TCP Performance Parameters

SUMMARY STEPS

1. **show tcp** [*line-number*] [**tcb** *address*]
2. **show tcp brief** [**all** | **numeric**]
3. **debug ip tcp transactions**
4. **debug ip tcp congestion**

DETAILED STEPS

Step 1 **show tcp** [*line-number*] [**tcb** *address*]

Displays the status of TCP connections. The arguments and keyword are as follows:

- *line-number*—(Optional) Absolute line number of the Telnet connection status.
- **tcb**—(Optional) Transmission control block (TCB) of the Explicit Congestion Notification (ECN)-enabled connection.
- *address*—(Optional) TCB hexadecimal address. The valid range is from 0x0 to 0xFFFFFFFF.

The following sample output from the **show tcp tcb** command displays detailed information about an ECN-enabled connection that uses a hexadecimal address format:

Example:

```
Device# show tcp tcb 0x62CD2BB8

Connection state is LISTEN, I/O status: 1, unread input bytes: 0
Connection is ECN enabled
Local host: 10.10.10.1, Local port: 179
Foreign host: 10.10.10.2, Foreign port: 12000
Enqueued packets for retransmit: 0, input: 0 mis-ordered: 0 (0 bytes)
Event Timers (current time is 0x4F31940):
Timer          Starts    Wakeups            Next
Retrans             0          0             0x0
TimeWait            0          0             0x0
AckHold             0          0             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0
iss:          0 snduna:          0 sndnxt:          0     sndwnd:      0
irs:          0 rcvnxt:          0 rcvwnd:       131072  delrcvwnd:      0
SRTT: 0 ms, RTTO: 2000 ms, RTV: 2000 ms, KRTT: 0 ms
minRTT: 60000 ms, maxRTT: 0 ms, ACK hold: 200 ms
Flags: passive open, higher precedence, retransmission timeout
TCB is waiting for TCP Process (67)
Datagrams (max data segment is 516 bytes):
Rcvd: 6 (out of order: 0), with data: 0, total data bytes: 0
Sent: 0 (retransmit: 0, fastretransmit: 0), with data: 0, total data
bytes: 0
```

Cisco Software Modularity

The following sample output from the **show tcp tcb** command displays a Software Modularity image:

Example:

```
Device# show tcp tcb 0x1059C10

Connection state is ESTAB, I/O status: 0, unread input bytes: 0
Local host: 10.4.2.32, Local port: 23
Foreign host: 10.4.2.39, Foreign port: 11000
VRF table id is: 0
Current send queue size: 0 (max 65536)
Current receive queue size: 0 (max 32768)  mis-ordered: 0 bytes
Event Timers (current time is 0xB9ACB9):
Timer          Starts    Wakeups            Next(msec)
Retrans             6          0                0
SendWnd             0          0                0
TimeWait            0          0                0
AckHold             8          4                0
KeepAlive          11          0          7199992
PmtuAger            0          0                0
GiveUp              0          0                0
Throttle            0          0                0
irs:    1633857851  rcvnxt: 1633857890  rcvadv: 1633890620  rcvwnd:  32730
iss:    4231531315  snduna: 4231531392  sndnxt: 4231531392  sndwnd:   4052
sndmax: 4231531392  sndcwnd:     10220
SRTT: 84 ms,  RTTO: 650 ms,  RTV: 69 ms,  KRTT: 0 ms
minRTT: 0 ms,  maxRTT: 200 ms, ACK hold: 200 ms
Keepalive time: 7200 sec, SYN wait time: 75 sec
Giveup time: 0 ms, Retransmission retries: 0, Retransmit forever: FALSE
State flags: none
Feature flags: Nagle
Request flags: none
Window scales: rcv 0, snd 0, request rcv 0, request snd 0
Timestamp option: recent 0, recent age 0, last ACK sent          0
```

```
Datagrams (in bytes): MSS 1460, peer MSS 1460, min MSS 1460, max MSS 1460
Rcvd: 14 (out of order: 0), with data: 10, total data bytes: 38
Sent: 10 (retransmit: 0, fastretransmit: 0), with data: 5, total data bytes: 76
Header prediction hit rate: 72 %
Socket states: SS_ISCONNECTED, SS_PRIV
Read buffer flags: SB_WAIT, SB_SEL, SB_DEL_WAKEUP
Read notifications: 4
Write buffer flags: SB_DEL_WAKEUP
Write notifications: 0
Socket status: 0
```

Step 2 **show tcp brief** [**all** | **numeric**]

(Optional) Displays addresses in IP format.

Use the **show tcp brief** command to display a concise description of TCP connection endpoints. Use the optional **all** keyword to display the status for all endpoints with addresses in a Domain Name System (DNS) hostname format. If this keyword is not used, endpoints in the LISTEN state are not shown. Use the optional **numeric** keyword to display the status for all endpoints with addresses in IP format.

Note If the **ip domain-lookup** command is enabled on the device, and you execute the **show tcp brief** command, the response time of the device to display the output will be very slow. To get a faster response, you should disable the **ip domain-lookup** command.

The following is sample output from the **show tcp brief** command while a user is connected to the system by using Telnet:

Example:

```
Device# show tcp brief

TCB       Local Address           Foreign Address        (state)
609789AC  Device.cisco.com.23     cider.cisco.com.3733   ESTAB
```

The following example shows the IP activity after the **numeric** keyword is used to display addresses in IP format:

Example:

```
Device# show tcp brief numeric

TCB           Local Address          Foreign Address     (state)
6523A4FC      10.1.25.3.11000        10.1.25.3.23         ESTAB
65239A84      10.1.25.3.23           10.1.25.3.11000      ESTAB
653FCBBC      *.1723 *.* LISTEN
```

Step 3 **debug ip tcp transactions**

Use the **debug ip tcp transactions** command to display information about significant TCP transactions such as state changes, retransmissions, and duplicate packets. The TCP/IP network isolated above the data link layer might encounter performance issues. The **debug ip tcp transactions** command can be useful in debugging these performance issues.

The following is sample output from the **debug ip tcp transactions** command:

Example:

```
Device# debug ip tcp transactions

TCP: sending SYN, seq 168108, ack 88655553
TCP0: Connection to 10.9.0.13:22530, advertising MSS 966
TCP0: state was LISTEN -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: state was SYNSENT -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: Connection to 10.9.0.13:22530, received MSS 956
TCP0: restart retransmission in 5996
TCP0: state was SYNRCVD -> ESTAB [23 -> 10.9.0.13(22530)]
```

```
TCP2: restart retransmission in 10689
TCP2: restart retransmission in 10641
TCP2: restart retransmission in 10633
TCP2: restart retransmission in 13384 -> 10.0.0.13(16151)]
TCP0: restart retransmission in 5996 [23 -> 10.0.0.13(16151)]
```

The following line from the **debug ip tcp transactions** command sample output shows that TCP has entered Fast Recovery mode:

Example:

```
fast re-transmit - sndcwnd - 512, snd_last - 33884268765
```

The following lines from the **debug ip tcp transactions** command sample output show that a duplicate acknowledgment is received when TCP is in Fast Recovery mode (first line) and a partial acknowledgment has been received (second line):

Example:

```
TCP0:ignoring second congestion in same window sndcwn - 512, snd_1st - 33884268765
TCP0:partial ACK received sndcwnd:338842495
```

Step 4 **debug ip tcp congestion**

Use the **debug ip tcp congestion** command to display information about TCP congestion events. The TCP/IP network isolated above the data link layer might encounter performance issues. The **debug ip tcp congestion** command can be used to debug these performance issues. The command also displays information related to variations in the TCP send window, congestion window, and congestion threshold window.

The following is sample output from the **debug ip tcp congestion** command:

Example:

```
Device# debug ip tcp congestion

*May 20 22:49:49.091: Setting New Reno as congestion control algorithm
*May 22 05:21:47.281: Advance cwnd by 12
*May 22 05:21:47.281: TCP85FD0C10: sndcwnd: 1472
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1475
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1478
*May 22 05:21:47.285: Advance cwnd by 9
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1487
*May 20 22:50:32.559: [New Reno] sndcwnd: 8388480 ssthresh: 65535 snd_mark: 232322
*May 20 22:50:32.559: 10.168.10.10:42416 <---> 10.168.30.11:49100 congestion window changes
*May 20 22:50:32.559: cwnd from 8388480 to 2514841, ssthresh from 65535 to 2514841
```

For Cisco TCP, New Reno is the default congestion control algorithm. However, an application can also use Binary Increase Congestion Control (BIC) as the congestion control algorithm. The following is sample output from the **debug ip tcp congestion** command using BIC:

Example:

```
Device# debug ip tcp congestion

*May 22 05:21:42.281: Setting BIC as congestion control algorithm
*May 22 05:21:47.281: Advance cwnd by 12
*May 22 05:21:47.281: TCP85FD0C10: sndcwnd: 1472
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1475
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1478
```

```
*May 22 05:21:47.285: Advance cwnd by 9
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1487
*May 20 22:50:32.559: [BIC] sndcwnd: 8388480 ssthresh: 65535 bic_last_max_cwnd: 0 last_cwnd: 8388480
*May 20 22:50:32.559: 10.168.10.10:42416 <---> 10.168.30.11:49100 congestion window changes
*May 20 22:50:32.559: cwnd from 8388480 to 2514841, ssthresh from 65535 to 2514841
*May 20 22:50:32.559: bic_last_max_cwnd changes from 0 to 8388480
```

Configuration Examples for TCP

Example: Verifying the Configuration of TCP ECN

The following example shows how to verify whether TCP ECN is configured:

```
Device# show running-config

Building configuration...
.
.
.
ip tcp ecn ! ECN is configured.
.
.
.
```

The following example shows how to verify whether TCP is ECN-enabled on a specific connection (local host):

```
Device# show tcp tcb 123456A

!Local host
!
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Enabled
Local host: 10.1.25.31, Local port: 11002
Foreign host: 10.1.25.34, Foreign port: 23
```

The following example shows how to display concise information about one address:

```
Device# show tcp brief

!
TCB          Local address            Foreign Address        (state)
609789C      Router.example.com.23      cider.example.com.3733    ESTAB
```

The following example shows how to enable IP TCP ECN debugging:

```
Device# debug ip tcp ecn
!
TCP ECN debugging is on
!
Device# telnet 10.1.25.31

Trying 10.1.25.31 ...
!
01:43:19: 10.1.25.35:11000 <---> 10.1.25.31:23   out ECN-setup SYN
```

```
01:43:21: 10.1.25.35:11000 <---> 10.1.25.31:23   congestion window changes
01:43:21: cwnd from 1460 to 1460, ssthresh from 65535 to 2920
01:43:21: 10.1.25.35:11000 <---> 10.1.25.31:23   in non-ECN-setup SYN-ACK
```

Before a TCP connection can use ECN, a host sends an ECN-setup SYN (synchronization) packet to a remote end that contains an Echo Congestion Experience (ECE) and Congestion window reduced (CWR) bit set in the header. Setting the ECE and CWR bits indicates to the remote end that the sending TCP is ECN capable, rather than an indication of congestion. The remote end sends an ECN-setup SYN-ACK (acknowledgment) packet to the sending host.

In this example the “out ECN-setup SYN” text means that a SYN packet with the ECE and CWR bit set was sent to the remote end. The “in non-ECN-setup SYN-ACK” text means that the remote end did not favorably acknowledge the ECN request and, therefore, the session is not ECN capable.

The following output shows that ECN capabilities are enabled at both ends. In response to the ECN-setup SYN, the other end favorably replied with an ECN-setup SYN-ACK message. This connection is now ECN capable for the rest of the session.

```
Device# telnet 10.10.10.10

Trying 10.10.10.10 ... Open
Password required, but none set
!
1d20h: 10.1.25.34:11003 <---> 10.1.25.35:23   out ECN-setup SYN
1d20h: 10.1.25.34:11003 <---> 10.1.25.35:23   in ECN-setup SYN-ACK
```

The following example shows how to verify that the hosts are connected:

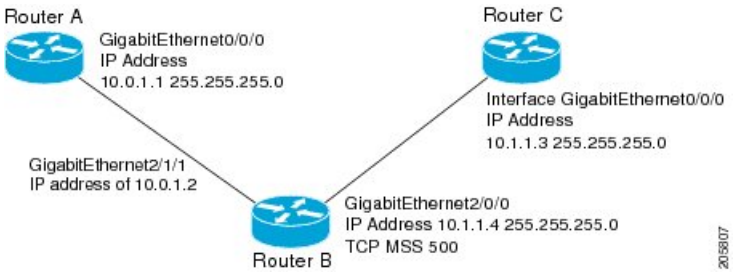
```
Device# show debugging
!
TCP:
  TCP Packet debugging is on
  TCP ECN debugging is on
!
Device# telnet 10.1.25.234
!
Trying 10.1.25.234 ...
!
00:02:48: 10.1.25.31:11001 <---> 10.1.25.234:23   out ECN-setup SYN
00:02:48: tcp0: O CLOSED 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 131072
00:02:50: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:02:50: cwnd from 1460 to 1460, ssthresh from 65535 to 2920
00:02:50: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 131072
00:02:54: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:02:54: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:02:54: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 131072
00:03:02: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:03:02: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:02: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 131072
00:03:18: 10.1.25.31:11001 <---> 10.1.25.234:23   SYN with ECN disabled
00:03:18: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:03:18: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:18: tcp0: O SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 131072
00:03:20: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:03:20: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:20: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
```

```
        OPTS 4 SYN  WIN 131072
00:03:24: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:03:24: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:24: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 131072
00:03:32: 10.1.25.31:11001 <---> 10.1.25.234:23   congestion window changes
00:03:32: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:32: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 131072
!Connection timed out; remote host not responding
```

Example: Configuring the TCP MSS Adjustment

The following example shows how to configure and verify the interface adjustment value for the example topology displayed in the figure below:

Figure 1: Example Topology for TCP MSS Adjustment



Configure the interface adjustment value on router B:

```
Router_B(config)# interface GigabitEthernet 2/0/0
Router_B(config-if)# ip tcp adjust-mss 500
```

Telnet from router A to router C with B having the Maximum Segment Size (MSS) adjustment configured:

```
Router_A# telnet 192.168.1.1

Trying 192.168.1.1... Open
```

Observe the debug output from router C:

```
Router_C# debug ip tcp transactions

Sep 5 18:42:46.247: TCP0: state was LISTEN -> SYNRCVD [23 -> 10.0.1.1(38437)]
Sep 5 18:42:46.247: TCP: tcb 32290C0 connection to 10.0.1.1:38437, peer MSS 500, MSS is 500
Sep 5 18:42:46.247: TCP: sending SYN, seq 580539401, ack 6015751
Sep 5 18:42:46.247: TCP0: Connection to 10.0.1.1:38437, advertising MSS 500
Sep 5 18:42:46.251: TCP0: state was SYNRCVD -> ESTAB [23 -> 10.0.1.1(38437)]
```

The MSS gets adjusted to 500 on Router B as configured.

The following example shows the configuration of a Point-to-Point Protocol over Ethernet (PPPoE) client with the MSS value set to 1452:

```
Device(config)# vpdn enable
Device(config)# no vpdn logging
Device(config)# vpdn-group 1
Device(config-vpdn)# request-dialin
```

```
Device(config-vpdn-req-in)# protocol pppoe
Device(config-vpdn-req-in)# exit
Device(config-vpdn)# exit
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# ip address 192.168.100.1.255.255.255.0
Device(config-if)# ip tcp adjust-mss 1452
Device(config-if)# ip nat inside
Device(config-if)# exit
Device(config)# interface ATM 0
Device(config-if)# no ip address
Device(config-if)# no atm ilmi-keepalive
Device(config-if)# pvc 8/35
Device(config-if)# pppoe client dial-pool-number 1
Device(config-if)# dsl equipment-type CPE
Device(config-if)# dsl operating-mode GSHDSL symmetric annex B
Device(config-if)# dsl linerate AUTO
Device(config-if)# exit
Device(config)# interface Dialer 1
Device(config-if)3 ip address negotiated
Device(config-if)# ip mtu 1492
Device(config-if)# ip nat outside
Device(config-if)# encapsulation ppp
Device(config-if)# dialer pool 1
Device(config-if)# dialer-group 1
Device(config-if)# ppp authentication pap callin
Device(config-if)# ppp pap sent-username sohodyn password 7 141B1309000528
Device(config-if)# ip nat inside source list 101 Dialer1 overload
Device(config-if)# exit
Device(config)# ip route 0.0.0.0.0.0.0.0 Dialer1
Device(config)# access-list permit ip 192.168.100.0.0.0.0.255 any
```

The following example shows the configuration of interface adjustment value for IPv6 traffic:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 0/0/0
Device(config)# ipv6 tcp adjust-mss 1452
Device(config)# end
```

Example: Configuring the TCP Application Flags Enhancement

The following output shows the flags (status and option) displayed using the **show tcp** command:

```
Device# show tcp
.
.
.
Status Flags: passive open, active open, retransmission timeout
 App closed
Option Flags: vrf id set
IP Precedence value: 6
.
.
.
SRTT: 273 ms, RTTO: 490 ms, RTV: 217 ms, KRTT: 0 ms
minRTT: 0 ms, maxRTT: 300 ms, ACK hold: 200 ms
```

Example: Displaying Addresses in IP Format

The following example shows the IP activity by using the **numeric** keyword to display the addresses in IP format:

```
Device# show tcp brief numeric

TCB           Local Address          Foreign Address     (state)
6523A4FC      10.1.25.3.11000        10.1.25.3.23         ESTAB
65239A84      10.1.25.3.23           10.1.25.3.11000      ESTAB
653FCBBC      *.1723 *.* LISTEN
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IP Application Services commands	IP Application Services Command Reference

Standards and RFCs

Standard/RFC	Title
RFC 793	Transmission Control Protocol
RFC 1191	Path MTU discovery
RFC 1323	TCP Extensions for High Performance
RFC 2018	TCP Selective Acknowledgment Options
RFC 2581	TCP Congestion Control
RFC 3168	The Addition of Explicit Congestion Notification (ECN) to IP
RFC 3782	The NewReno Modification to TCP's Fast Recovery Algorithm
RFC 4022	Management Information Base for the Transmission Control Protocol (TCP)

MIBs

MIB	MIBs Link
CISCO-TCP-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for TCP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for TCP

Feature Name	Releases	Feature Information
TCP Application Flags Enhancement	12.2(31)SB2 12.4(2)T	The TCP Applications Flags Enhancement feature enables the user to display additional flags with reference to TCP applications. There are two types of flags: status and option. The status flags indicate the status of TCP connections such as retransmission timeouts, application closed, and synchronized (SYNC) handshakes for listening. The additional flags indicate the state of set options such as whether a VPN routing and forwarding instance (VRF) is set, whether a user is idle, and whether a keepalive timer is running. The following command was modified by this feature: **show tcp**.

Feature Name	Releases	Feature Information
TCP Congestion Avoidance	12.3(7)T	The TCP Congestion Avoidance feature enables the monitoring of acknowledgment packets to the TCP sender when multiple packets are lost in a single window of data. Before this feature was introduced, the sender would exit Fast-Recovery mode, wait for three or more duplicate acknowledgment packets before retransmitting the next unacknowledged packet, or wait for the retransmission timer to start slowly. This delay could lead to performance issues. Implementation of RFC 2581 and RFC 3782 addresses the modifications to the Fast-Recovery algorithm that incorporates a response to partial acknowledgments received during Fast Recovery, improving performance in situations where multiple packets are lost in a single window of data. This feature is an enhancement to the existing Fast Recovery algorithm. No commands are used to enable or disable this feature. The output of the **debug ip tcp transactions** command monitors acknowledgment packets by displaying the following conditions: • TCP entering Fast Recovery mode. • Duplicate acknowledgments being received during Fast Recovery mode. • Partial acknowledgments being received. The following command was modified by this feature: **debug ip tcp transactions**.
TCP Explicit Congestion Notification	12.3(7)T	The TCP Explicit Congestion Notification (ECN) feature allows an intermediate router to notify end hosts of impending network congestion. It also provides enhanced support for TCP sessions associated with applications such as Telnet, web browsing, and transfer of audio and video data, that are sensitive to delay or packet loss. The benefit of this is the reduction of delay and packet loss in data transmissions. The following commands were introduced or modified by this feature: **debug ip tcp ecn**, **ip tcp ecn**, **show debugging**, **show tcp**.
TCP MIB for RFC4022 Support	12.2(33)XN	The TCP MIB for RFC 4022 Support feature introduces support for RFC 4022, *Management Information Base for the Transmission Control Protocol (TCP)*. RFC 4022 is an incremental change of the TCP MIB to improve the manageability of TCP. There are no new or modified commands for this feature.

Feature Name	Releases	Feature Information
TCP MSS Adjust	12.2(4)T 12.2(8)T 12.2(18)ZU2 12.2(28)SB 12.2(33)SRA 12.2(33)SXH 15.0(1)S	The TCP MSS Adjust feature enables the configuration of the maximum segment size (MSS) for transient packets that traverse a device, specifically TCP segments in the SYN bit set. In 12.2(4)T, this feature was introduced. In 12.2(8)T, the command that was introduced by this feature was changed from **ip adjust-mss** to **ip tcp adjust-mss**. In 12.2(28)SB and 12.2(33)SRA, this feature was enhanced to be configurable on subinterfaces. The following command was introduced by this feature: **ip tcp adjust-mss**.
TCP Show Extension	12.2(31)SB2 12.4(2)T	The TCP Show Extension feature introduces the capability to display addresses in IP format instead of hostname format and to display the VRF table associated with the connection. The following command was modified by this feature: **show tcp brief**.
TCP Window Scaling	12.2(8)T 12.2(31)SB2	The TCP Window Scaling feature adds support for the Window Scaling option in RFC 1323. A larger window size is recommended to improve TCP performance in network paths with large bandwidth, long-delay characteristics that are called Long Fat Networks (LFNs). This TCP Window Scaling enhancement provides that support. The following command was introduced or modified by this feature: **ip tcp window-size**.
TCP Keepalive Timer	15.2(4)M	The TCP Keepalive Timer feature introduces the capability to identify dead connections between multiple routing devices. The following command was introduced or modified by this feature: **ip tcp keepalive**.